AdClick prediction using improved AdaGrad Algorithm

Darshankumar Kapadiya
Institute of Technology
Nirma University
Ahmedabad, India
21mcec02@nirmauni.ac.in

Yagnesh Bhadiyadra
Institute of Technology
Nirma University
Ahmedabad, India
https://orcid.org/0000-0002-8017-9535

***Abstract*—Advertisements are the major source of income for the trillion-dollar IT industry around the world. Users' clicks on a particular hyperlink decide the response of a particular App. Different search engines like google/Bing and advertisers depends on it, so Predicting the CTR(click through rate) is a very important part of the optimization of the particular application. We in this paper present a novel method integrating Logistic Regression with an improved version of the Adaptive Gradient Descent algorithm, in which we change learning rate according to the gradient of the learnable parmeters with respect to the cost function and get the results. We find that our algorithm gets the desired accuracy in lesser number of Epochs rather than Regular Logistic Regression.**

***Index Terms*—AdClick Prediction, CTR, gradient descent, Improved adagrad, linear regression.**

I. Introduction

IT industry is growing like anything and no one is stopping the revenue generation happening by IT giants. According to Wikipedia, Samsung only accounts for 17 % of the GDP of the South Korea. The web applications are one of the pivotal things in improving an IT Company's revenue especially if the company is like Google, whose major revenue comes from brands paying them to display their ads on the search engine when a user searches for a keyboard. Click through rate is number of impressions that resulted in a click. The formula for CTR (Click-through Rate) is given below:

$$CTR = \frac{\text{Number of click throughs}}{\text{Number of impressions}} * 100(\%) \tag{1}$$

So how does the CTR effects the revenue generated by companies like Google, Facebook (upto some extent its revenue is generated by ads)? There is a very simple answer to this [2]. Different advertisers approach these search engines with their ads and the bidding amount to display their ads. First the engine calculates the probability of a click given the features:

- Ad
- context
- User

$Pr(Click/user, ad, context)$: The conditional probability is calculated and the bidding amount is multiplied with that which is the final revenue:

$$\begin{aligned}\text{Revenue Generated} = \Pr(\text{Click / User, Ad, Context}) \\ *\text{Bidding Amount}\end{aligned} \tag{2}$$

The rest of the paper is organized as follows : section II described the related work in the field of AdClick prediction and in the area of Gradient Descent optimization technique. section III describes our proposed work and section IV concludes the paper and discusses the future scope.

II. Related Work

There is a lot of work in the field of AdClick prediction. However a paper by Google Employees [3] in 2013 suggested FTRL-proximal online learning algorithm and the use of per-coordinate learning rate which improves traditional supervised learning algorithm. There is some work done also in the area of Gradient Descent based optimization techniques. Janis Kuper et al. [4]in 2015 proposed a novel technique which tries to address the shortcomings of Hadoop MapReduce technique in parallel system. MapReduce fails to provide good scalability to the system when it comes to batch gradient descent, because it needs to read the entire dataset at once, hence scalability with respect to entire dataset is poor. Stochastic Gradient Descent is also an option to explore here. However it also hard to parallelise due to inherent sequential nature. The authors here proposed an algorithm based on one-sided asynchronous mode of Inter-Process Communication.

Another work that has been done has tried to improve the Adaptive Gradient Descent algorithm to improve the convergence rate of the same. The AdaGrad (Adaptive Gradient Descent) algorithm works a little differently than state-of-the-art Gradient Descent algorithm. It keeps track of the sum of gradient squared and uses that to adapt the gradient in different directions unlike sum in case of normal Gradient Descent. It is used for data with some sparse and some dense parameters. Sparse parameter converge slowly than dense parameters, and thus we use this algorithm to change learning rate dynamically based on convergence with respect to various parameters. The overview of the AdaGrad algorithm [5] is depicted in Figure 1. In the figure, we can see that white trajectory is for the AdaGrad algorithm while cyan trajectory is for State-of-the-art gradient descent algorithm. In Machine Learning optimization,

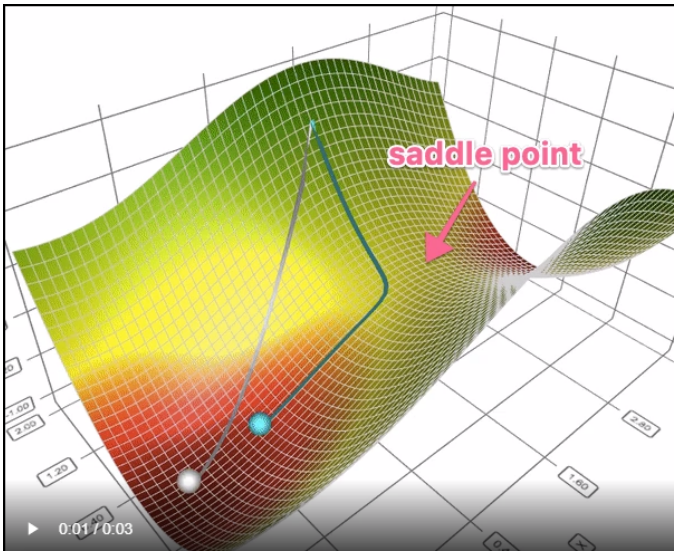

Figure 1. Working of AdaGrad Algorithm

Input:
nb_epochs ← number of epochs
lr ← learning rate

BEGAIN:
smooth_t ← smooth term
$s \leftarrow 0$
epochs ← 0
while *epochs* < *nb_epochs*
do $g \leftarrow$ get gradients of θ
$loss \leftarrow J(\theta)$
$s \leftarrow s +$ length of g
$\theta \leftarrow \theta - lr * g / (s + smooth\_t)$
$epochs \leftarrow epochs + 1$
return *loss*

END

Figure 2. Improved AdaGrad Algorithm

some features are very sparse. The average gradient for sparse features is usually small, so such features get trained at a much slower rate with state-of-the-art gradient descent. One way to address this is to set different learning rates for each feature, but this gets too messy very quickly. AdaGrad addresses this problem using the following idea: the more you have updated a feature already, the less you will update it in the future, thus giving a chance for the others features (for example, the sparse features) to catch up [6]. In visual terms, how much you have updated this feature is to say how much you have moved in this dimension, and this notion is captured by the cumulative sum of gradient squared.

N. Zhang, D. Lei, and J. F. Zhao in 2018 proposed a work that focused on improving the AdaGrad algorithm depicted in Figure 1. The algorithm takes the length of gradient into consideration rather than the squared gradient as with AdaGrad algorithm and they get faster convergance than the same algorithm as well. They test their approach both on the Reuters dataset and the IMDB dataset. The result shows that their approach has a more stable convergence process and can reduce overfitting in multiple epoch.

As we can see, most of the articles either talk about how to modify the existing gradient descent to improve the same, or they talk about a specific application using the gradient descent to solve a business problem, like AdClick Prediction. So, we propose bridging the gap between two research dimensions by introducing AdClick prediction with Logistic Regression with improved AdaGrad algorithm.

III. Proposed Experiment

First we take the AdClick prediction data-set and do the Exploratory Data Analysis. We see that it has no null or missing values, and it has equal class distribution of the two output classes (either clicked on the particular ad (0) or not clicked on the particular ad (1). It has six features that are important to be considered for the classification :

- Daily time spent on site
- Age
- Daily internet usage
- Gender
- Clicked on Ad (or not)

The experiment proposed by us takes into account adclick prediction with Logistic Regression but the Gradient Descent technique used for optimization is slightly modified as described above. The data for the AdClick prediction is taken from [2]. The Logistic Regression will be of mini-batch type and we have kept the batch size fixed at 10, for each epoch. The Improved AdaGrad algorithm is described in figure 2.

As we can see the improved AdaGrad algorithm changes the way the updation of weights is done. We can see that it takes the length of the gradient and divided them with the learning rate at each iteration for each feature. This way we are dynamically changing the learning rate with respect to how a particular feature is affecting the cost function. We have run the Improved AdaGrad algorithm, State-of-the-art gradient descent algorithm and have compared the results, for different number of epochs. The metric for comparing the performance is accuracy of the classification on the test data. We can see that the AdaGrad algorithm gives better results for very lesser number of epochs rather than the regular AdaGrad algorithm and we are getting close to 95 % accuracy in AdaGrad whereas for the same number of epochs, state-of-the-art gradient descent algorithm is having close to 92% accuracy. And, in the further iterations, we also see that AdaGrad most of the time outperforms State-of-the-art gradient descent in terms of accuracy. Figure 3, 4, and 5 shows the results in graphical format (AdaGrad in these figures refer to improved AdaGrad).

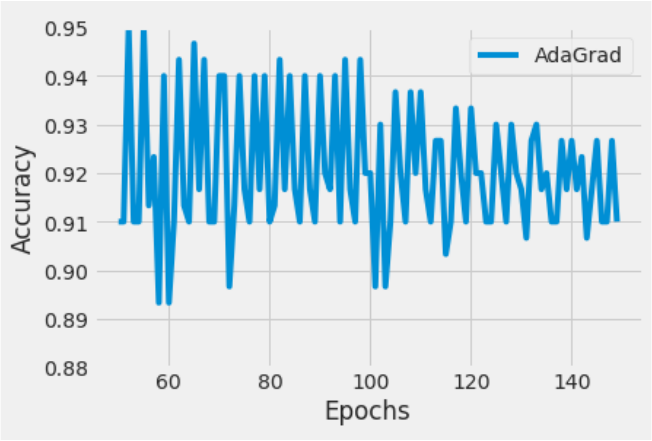

Figure 3. Improved AdaGrad Algorithm

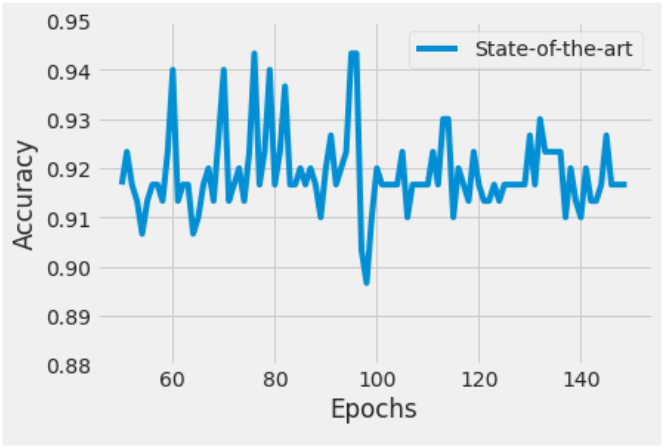

Figure 4. State of the art Gradient Descent

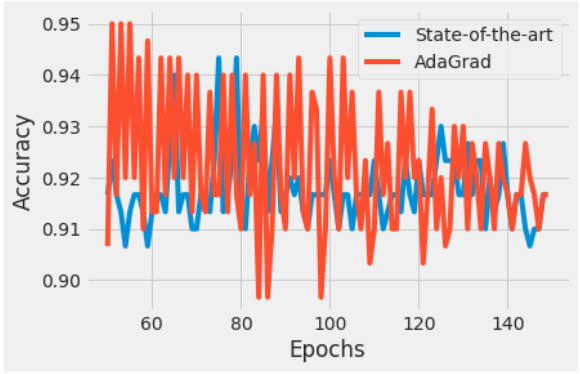

Figure 5. Overlapping of results both the algorithms

IV. Conclusion and Future Scope

Gradient Descent is a very important technique when it comes to optimization problems. However, we some modifications in the existing technique can result in better results for the given problem. Here we try to exploit one of the possible modifications of the gradient descent technique and we see that we can get a good improvement over the existing algorithm in terms of faster convergence and getting better accuracy.In future, we can try various other variations of Gradient Descent. In this work we tried to change the learning rate of the individual features, we could play with stochastic gradient descent which is also one of the most used technique when dataset is too large, and improving learning rate in those algorithms to get better classification results. However, Particle Swarm optimization and Genetic algorithms are sometimes better alternatives to the Gradient Descent because they do not need the information about the gradients and can be used like a black box itself.

References

[1] Zhang, N. Lei, D. Zhao, J.F.. (2018). An Improved Adagrad Gradient Descent Optimization Algorithm. 2359-2362. 10.1109/CAC.2018.8623271.

[2] S. Dodeja, "Ad Click Prediction: What, why, and how?", *Medium*, 2021. [Online]. Available: https://medium.com/geekculture/ad-click-prediction-what-why-and-how-bee259ddb05c. [Accessed: 08- Dec- 2021]

[3] H. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. Hrafnkelsson, T. Boulos and J. Kubica, "Ad click prediction", *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013* [Online]. Available: https://doi.org/10.1145/2487575.2488200. [Accessed: 09- Dec- 2021]

[4] Keuper, J. and Pfreundt, F., 2015. Asynchronous parallel stochastic gradient descent. Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, [online] Available at: ¡https://dl.acm.org/doi/10.1145/2834892.2834893¿ [Accessed 11 December 2021].

[5] Jiang, L., 2021. *A Visual Explanation of Gradient Descent Methods (Mometum, AdaGrad, RMSProp, Adam)*. [online] Medium. Available at: ¡https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c¿ [Accessed 11 December 2021].

[6] Asquero. 2021. *Advantages and Disadvantages of Stochastic Gradient Descent | Asquero*. [online] Available at: ¡https://www.asquero.com/article/advantages-and-disadvantages-of-stochastic-gradient-descent/¿ [Accessed 11 December 2021].